UNITED STATES

PATENT APPLICATION

for

Operation Control for Data Types

NCR Docket No. 11351

submitted by

Gregory H. Milby

on behalf of

Teradata

a Division of NCR Corporation

Dayton, Ohio

Prepared by

Bradley S. Bowling
Reg. No. 52,641

Correspond with

John D. Cowart
Reg. 38,415
Teradata Law IP, WHQ-4W
NCR Corporation
1700 S. Patterson Blvd.
Dayton, OH 45479-0001
(858) 485-4903 [Voice]
(858) 485-2581 [Fax]

HOU03:950220.1

Express Mail No. EV339222620US
Date: March 19, 2004

Activation of Native Operations for Distinct-User Defined Types

Background

[0001] The ANSI SQL:1999 standard introduced support for distinct and structured user-defined types (UDTs). The standard enables developers of distinct and structured UDTs to define the behavior of the types with respect to type comparisons, casting functionality, and transforms (*i.e.* input and output formatting).

Summary

[0002] In general, in one aspect, the invention features a method of controlling operations that may be performed on a user-defined type (UDT) in a database system. The UDT is derived from an underlying type that has a set of underlying operations. The method includes creating the UDT and activating zero or more underlying operations for the UDT.

[0003] Implementations of the invention may include one or more of the following. The UDT may be a distinct data type. Creating the UDT may include accepting a CREATE TYPE query. The CREATE TYPE query may include a system operators clause. The system operators clause may include an operator list, which may include zero or more operator elements, where each operator element corresponds to an underlying operation. Activating the underlying operations may include, for each operator element listed in the operator list, activating the corresponding underlying operation for the UDT. The operator list may include an ALL element. Activating the underlying operations may include activating all underlying operations for the UDT. Activating the underlying operations may include accepting an ALTER TYPE query that includes a system operators clause. The system operators clause may include an operator list. The operator list may include zero or more operator elements, where each operator element corresponds to an underlying operation. Activating the underlying operations may include, for each operator element listed in the operator list, activating the corresponding underlying operation for the UDT. Activating the underlying operations may include recording, in a data dictionary, the activated underlying operations for the UDT. One or more UDT columns may be adapted to store UDT values. The method may include accepting a query including a query operator that takes one or more operands, where one or more of the operands are UDT columns. The method may include determining whether the query operator is activated for the UDT of each UDT column, and if it is, performing the operation.

[0004] In general, in another aspect, the invention features a computer program, stored on a tangible storage medium, for use in controlling operations that may be performed on a user-defined type (UDT) in a database system. The UDT is derived from an underlying type that has a set of underlying operations. The computer program includes executable instructions that cause a computer to create the UDT and activate zero or more underlying operations for the UDT.

[0005] In general, in another aspect, the invention features a database system that includes a massively parallel processing system. The massively parallel processing system includes one or more nodes, a plurality of CPUs, each of the one or more nodes providing access to one or more CPUs, a plurality of data storage facilities, each of the one or more CPUs providing access to one or more data storage facilities, and a process for execution on the massively parallel processing system for controlling operations that may be performed on a user-defined type (UDT) in the database system. The UDT is derived from an underlying type that has a set of underlying operations. The process includes creating the UDT and activating zero or more underlying operations for the UDT.

Brief Description of the Drawings

[0006] Fig. 1 is a block diagram of a node of a database system.

[0007] Fig. 2 is a block diagram of a parsing engine.

[0008] Fig. 3 is a flow chart a flow chart of a parser.

[0009] Fig. 4 is a flow chart of a system for creating a UDT.

[0010] Fig. 5 is a flow chart of a system for modifying activated operators for a UDT.

[0011] Fig. 6 is a flow chart of a system for determining whether a operator is activated.

Detailed Description

[0012] The techniques for statistically representing skewed data disclosed herein have particular application, but are not limited, to large databases that might contain many millions or billions of records managed by a database system ("DBS") 100, such as a Teradata Active Data Warehousing System available from NCR Corporation. Figure 1 shows a sample architecture for one node 105_1 of the DBS 100. The DBS node 105_1 includes one or more processing modules $110_{1...N}$, connected by a network 115, that manage the storage and retrieval of data in data-storage facilities $120_{1...N}$. Each of the processing modules $110_{1...N}$ may be one or more

physical processors or each may be a virtual processor, with one or more virtual processors running on one or more physical processors.

[0013] For the case in which one or more virtual processors are running on a single physical processor, the single physical processor swaps between the set of N virtual processors.

[0014] For the case in which N virtual processors are running on an M-processor node, the node's operating system schedules the N virtual processors to run on its set of M physical processors. If there are 4 virtual processors and 4 physical processors, then typically each virtual processor would run on its own physical processor. If there are 8 virtual processors and 4 physical processors, the operating system would schedule the 8 virtual processors against the 4 physical processors, in which case swapping of the virtual processors would occur.

[0015] Each of the processing modules $110_{1...N}$ manages a portion of a database that is stored in a corresponding one of the data-storage facilities $120_{1...N}$. Each of the data-storage facilities $120_{1...N}$ includes one or more disk drives. The DBS may include multiple nodes $105_{2...O}$ in addition to the illustrated node 105_1, connected by extending the network 115.

[0016] The system stores data in one or more tables in the data-storage facilities $120_{1...N}$. The rows $125_{1...Z}$ of the tables are stored across multiple data-storage facilities $120_{1...N}$ to ensure that the system workload is distributed evenly across the processing modules $110_{1...N}$. A parsing engine 130 organizes the storage of data and the distribution of table rows $125_{1...Z}$ among the processing modules $110_{1...N}$. The parsing engine 130 also coordinates the retrieval of data from the data-storage facilities $120_{1...N}$ in response to queries received from a user at a mainframe 135 or a client computer 140. The DBS 100 usually receives queries and commands to build tables in a standard format, such as SQL.

[0017] In one implementation, the rows $125_{1...Z}$ are distributed across the data-storage facilities $120_{1...N}$ by the parsing engine 130 in accordance with their primary index. The primary index defines the columns of the rows that are used for calculating a hash value. The function that produces the hash value from the values in the columns specified by the primary index is called the hash function. Some portion, possibly the entirety, of the hash value is designated a "hash bucket." The hash buckets are assigned to data-storage facilities $120_{1...N}$ and associated

processing modules $110_{1...N}$ by a hash bucket map. The characteristics of the columns chosen for the primary index determine how evenly the rows are distributed.

[0018] In one example system, the parsing engine 130 is made up of three components: a session control 200, a parser 205, and a dispatcher 210, as shown in Fig. 2. The session control 200 provides the logon and logoff function. It accepts a request for authorization to access the database, verifies it, and then either allows or disallows the access.

[0019] Once the session control 200 allows a session to begin, a user may submit a SQL request, which is routed to the parser 205. As illustrated in Fig. 3, the parser 205 interprets the SQL request (block 300), checks it for proper SQL syntax (block 305), evaluates it semantically (block 310), and consults a data dictionary to ensure that all of the objects specified in the SQL request actually exist and that the user has the authority to perform the request (block 315). Finally, the parser 205 runs an optimizer (block 320), which develops the least expensive plan to perform the request.

[0020] Fig. 4 shows an example system for creating a UDT based on an underlying data type. In one example system where the UDT is based on a single underlying type, the UDT is called a distinct UDT. In general, the UDT is created with the following query:

```
CREATE TYPE <distinct type name>
        [ AS <representation> ]
        [ <system operators clause> ]
        [ <reference type specification> ]
        [ <cast option> ]
        [ <method specification list> ]
```

The bracketed clauses in the query are optional. One example system provides default functionality for clauses not specified in the CREATE TYPE query based on the type specified in the representation clause (*e.g.,* the underlying-data type). For example, the system may receive the following query:

```
CREATE TYPE Euro AS DECIMAL(8,2);
```

[0021] The system receives the query (block 405), and in response, it creates a new UDT based on the underlying-data type DECIMAL(8,2) (block 410). The system generates the following default functionality for the Euro:

1. type comparison (*e.g.*, the comparison of two Euro's behaves the same as the comparison of two DECIMAL(8,2)'s) (block 415);
2. casting functionality (*e.g.*, Euro's may be cast into DECIMAL(8,2)'s and vice-versa) (block 420); and
3. import/export formatting (*e.g.*, Euro's are imported or exported as DECIMAL(8,2)'s) (block 425).

[0022] In the example query above, no default functionality was specified for the system operators clause. In one implementation of the system, no underlying operations are activated for the Euro when the system operators clause is not specified. In other example queries, one or more underlying operations are enabled for the UDT with the following general syntax for the system operators clause:

<system operators clause> ::= OPERATORS <left paren> <operator list> <right paren>

where the operator list has the following form:

<operator list> :: = ALL | { <operator element> [{<comma> <operator element>}...] }

Using "ALL" in the operator list enables all underlying operations for the UDT (*e.g.*, all valid operators for the underlying type are enabled for the UDT). Other example operator lists specify one or more underlying operations. In response to such operator lists, the system enables the underlying operations enumerated in the operator list for the UDT.

[0023] For example, the system may receive the following query:

CREATE TYPE Euro AS DECIMAL(8,2) OPERATORS (+, -);

The system receives the query (block 405), creates the Euro UDT (block 410), and generates the default functionality described above (blocks 415-425). The system also enables the operations associated with the "+" and "-" underlying operators for the Euro UDT (block 430). In this

example system, when a Euro is the operand of a "+" or "-" operator, the system treats each Euro operand as a DECIMAL(8,2).

[0024] In the example above the system enabled numeric operations for the UDT. In other examples, the system enables other operators or functions for the UDT. For example, consider the following query:

CREATE TYPE Name AS VARCHAR(40) OPERATORS (CONCAT, SUBSTRING);

The system receives the query (block 405), creates the Name UDT (block 410), and generates the default functionality described above (blocks 415-425). The system also enables the underlying operations associated with the "CONCAT" and "SUBSTRING" operators for the Name UDT (block 430). In this example system, when a Name is an argument to the "CONCAT" or "SUBSTRING" functions, the functions treat the Name as a VARCHAR(40).

[0025] In one example system, UDT definitions, including enabled underlying operations described by functions and operators, are stored in a data dictionary, allowing the system to add, retrieve, or modify UDT definitions.

[0026] In the examples above, underlying operators were enabled for UDTs using CREATE TYPE queries. In other implementations, the underlying operations may be enabled after the UDT is created. For example, the system in Fig. 5 alters the enabled underlying operators for a UDT. The system receives an ALTER TYPE query (block 505) and modifies the enabled operators for the UDT (block 510). An example query to modify the enabled underlying operations is:

ALTER TYPE Euro OPERATORS (+, -, *);

The system receives the query (block 505) and modifies the enabled underlying operations for the Euro UDT so that the operations associated with the "+," "-," and "*" operators are enabled and the other operators are disabled (block 510).

[0027] In addition to creating and modifying UDTs, one example DBS 100 evaluates queries that include one or more operators with one or more UDT operands, or queries that include one or functions with one or more UDT arguments. An example system for determining if the operation is permitted for the UDT is shown in Fig. 6. The system receives an operator with one

or more UDT operands or a function with one or more UDT arguments (block 605). The system then determines if the operator or function is activated for each UDT operand/argument (block 610). In one implementation, the system references the UDT definition in the data dictionary to determine if the operation represented by the operator or function is enabled for the UDT. If the operator or function is enabled for each UDT operand or argument, the system returns "TRUE" (block 615), otherwise the system returns "FALSE" (block 620).

[0028] For example, assume that an Orders table has the following definition:

```
CREATE TABLE Orders (ID INT, Subtotal Euro, Tax Euro);
```

and, further assume that the system receives the following query:

```
SELECT Subtotal + Tax FROM Orders where ID=1;
```

The system receives the "+" operator and two references to the Euro UDT (block 605). In certain implementations, the system receives only a single reference to a UDT that is used multiple times as an operand to an operator or as a argument to a function. The system determines if the operation associated with the "+" operator is enabled for the Euro UDT (block 610), and, because it is, the system returns "TRUE" (block 615).

[0029] The foregoing description of the preferred embodiment of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.